Computer Systems
Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2023.035570
Article

Question-Answering Pair Matching Based on Question Classification and Ensemble Sentence Embedding

Jae-Seok Jang[1] and Hyuk-Yoon Kwon[2,*]

[1]Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul, 01811, Korea
[2]Department of Industrial Engineering/Graduate School of Data Science, Seoul National University of Science and Technology, Seoul, 01811, Korea
*Corresponding Author: Hyuk-Yoon Kwon. Email: hyukyoon.kwon@seoultech.ac.kr
Received: 26 August 2022; Accepted: 03 November 2022

Abstract: Question-answering (QA) models find answers to a given question. The necessity of automatically finding answers is increasing because it is very important and challenging from the large-scale QA data sets. In this paper, we deal with the QA pair matching approach in QA models, which finds the most relevant question and its recommended answer for a given question. Existing studies for the approach performed on the entire dataset or datasets within a category that the question writer manually specifies. In contrast, we aim to automatically find the category to which the question belongs by employing the text classification model and to find the answer corresponding to the question within the category. Due to the text classification model, we can effectively reduce the search space for finding the answers to a given question. Therefore, the proposed model improves the accuracy of the QA matching model and significantly reduces the model inference time. Furthermore, to improve the performance of finding similar sentences in each category, we present an ensemble embedding model for sentences, improving the performance compared to the individual embedding models. Using real-world QA data sets, we evaluate the performance of the proposed QA matching model. As a result, the accuracy of our final ensemble embedding model based on the text classification model is 81.18%, which outperforms the existing models by 9.81%~14.16% point. Moreover, in terms of the model inference speed, our model is faster than the existing models by 2.61~5.07 times due to the effective reduction of search spaces by the text classification model.

Keywords: Question-answering; text classification model; data augmentation; text embedding

1 Introduction

As the amount of data rapidly increases, it requires lots of effort and time to find data that satisfies a given requirement from large-scale data sets. The QA system [1,2] finds answers to a given question. The necessity of automatically finding answers is also increasing because it is very important and challenging from the large-scale QA data sets. Existing studies for the QA matching models can be largely classified

into 1) QA pair matching [3,4], 2) machine reading-based models [5,6], and 3) knowledge bases to answer questions (KBQA) [7–11]. First, for QA pairs matching, there have been methods [3,4] to choose the most similar question to a given question from the question and answer datasets. It has been mainly performed on the entire data sets or dataset within a category the question writer manually specifies. Second, for the machine reading-based model, there have been methods [5,6] to generate answers to questions from a set of ground truth documents containing both questions and answers. Third, for KBQA, there have been methods to find answers from the knowledge base (KB) structures that are constructed based on predefined templates of the questions [7,8]. Other methods have embedded the questions into KB structures through neural network models and found the answers from the KB structures [9,10].

In this paper, we deal with the QA pair matching approach. The existing studies for the approach are performed on the entire dataset or datasets within a category that the question writer manually specifies. Existing methods are time consuming and less accurate because the recommended answer for a given question is selected from the entire datasets without the classification of target datasets. To overcome the limitations of existing studies, we aim to automatically find the category to which the question belongs by employing the text classification model and to find the answer corresponding to the question within the category.

The proposed QA matching system consists of the following components: 1) text classification, which allows to accurately choose the category for the question, and 2) sentence embedding, which allows finding the most relevant questions in the category. First, we adopt a text classification model to improve the performance and usability of QA systems. That is, by choosing the category to which a given question belongs using the text classification model, we can effectively improve the accuracy of QA matching and reduce the search space. There have been lots of research efforts for text classification including machine learning models, such as support vector machine (SVM) [12], k-nearest neighbor (KNN) [13], Corner Classification Network [13], and Naïve Bayes classifier [14], neural network-based models such as convolutional neural network (CNN) [15] and recurrent neural network (RNN) [16], and transformer-based models such as bidirectional encoder representations from transformers (BERT) [17]. In this paper, instead of proposing new text classification models, we apply the various text classification models tailored to a real-world dataset collected from a mobile application in South Korea. To this end, we adopt a fully trained model that combines a long short-term memory (LSTM) neural network, a type of recurrent neural network that solves the gradient loss problem and a CNN neural network (LSTM-CNN) [18], and four Korean pre-trained models, KoBERT[1], KoELECTRA[2], Multilingual BERT[3], and KR-BERT-MEDIUM[4]. To improve the performance of each text classification model, we adopt text data augmentation techniques. In the classification problem, because data imbalance for a specific class occurs frequently, text data augmentation techniques have been presented [19–21]. Text data augmentation creates new texts for the classes having insufficient data so that the characteristics of the existing texts are maintained [19]. Fig. 1 shows a data imbalance occurred on the actual data sets used in this study, i.e., questions collected from a mobile application for entrance exams and careers. This shows that a class named ‘regular decision’, which is the smallest, has data sets about 12 times less than another class named ‘study method’, which is the largest. The representative text data augmentation includes easy data augmentation (EDA) [20] and back-translation [21]. In this paper, we adopted EDA [20] and Word2Vec-based data augmentation (WDA) by extending EDA based on the Word2Vec embedding model. Second, after selecting the category to which a given question belongs, we find the most relevant questions for the

[1] https://github.com/SKTBrain/KoBERT.

[2] https://github.com/monologg/KoELECTRA.

[3] https://github.com/google-research/bert/blob/master/multilingual.md.

[4] https://github.com/snunlp/KR-BERT-MEDIUM.

question from the questions in the category. To effectively measure the similarity between questions, we devise an ensemble sentence embedding model by combining well-performed individual embedding models. For this, we consider multiple individual embedding models, i.e., term frequency-inverse document frequency (TF-IDF) [22], Word2Vec [23], and Doc2Vec [24], and apply them to the unit of the sentence, i.e., for the question, by adopting the distance metrics in the embedded spaces. Based on the performance evaluation, we choose effective individual embedding models and present a method to combine them in measuring the similarity between sentences. Finally, using real-world datasets, we evaluate the performance of the proposed QA pair matching method. As a result, we show the effectiveness of each component of the proposed method and confirm that the proposed model outperforms the existing QA models.

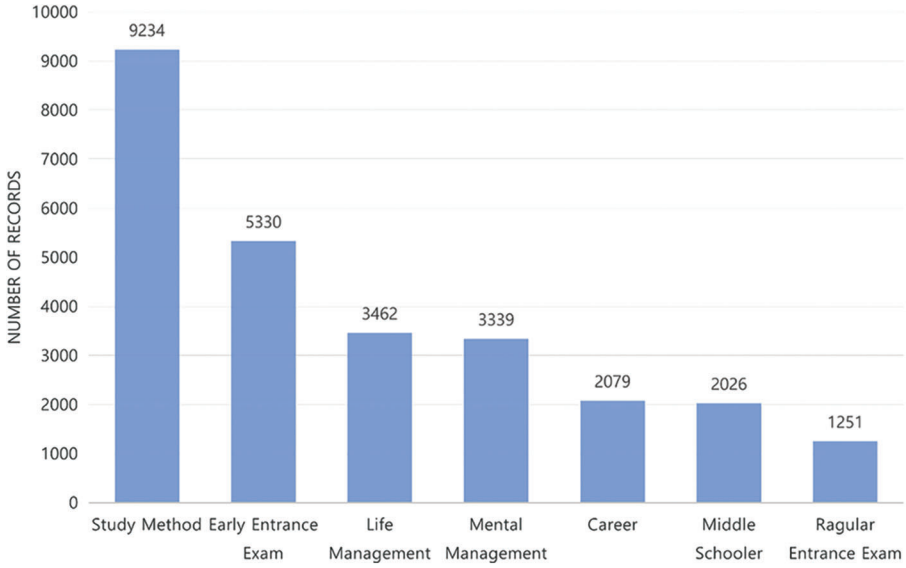

Figure 1: Distribution of questions for entrance exams and careers by the category

Specifically, in terms of the text classification model, KoBERT shows the best F1-score. In terms of the data augmentation, we confirm a clear performance improvement in the fully trained model, i.e., LSTM-CNN, while performance improvement in the BERT-based pre-trained models is limited. In terms of sentence embedding, we confirm that Word2Vec and TF-IDF clearly outperform Doc2Vec. Therefore, we present an ensemble sentence embedding model based on them, indicating its superiority compared to all the individual embedding models. As a result, the accuracy of our final ensemble embedding model based on the text classification model is 81.18%, which outperforms the existing models by 9.81%~14.16%. Moreover, in terms of the model inference speed, our model is faster than the existing models by 2.61~5.07 times due to the effective reduction of search spaces by the text classification model. We summarize the contributions of our paper as follows: 1) To effectively choose the most relevant category for a given question in QA matching, we employ the text classification model in QA pair matching. Due to the text classification model, we can effectively reduce the search space for finding the answers to a given question. Therefore, the proposed QA pair matching model improves the accuracy and significantly reduces the model inference time compared to the existing QA matching models. 2) To further improve the performance of finding the relevant questions in each category, we present an ensemble sentence embedding model combining individual models, indicating the performance improvement compared to the individual embedding models. 3) Using real-world QA datasets, we evaluate the performance of the

proposed QA matching model. As a result, we show the effectiveness of each component in the proposed method, and finally, confirm that the proposed model outperforms the existing QA matching models. The proposed framework focuses on the QA matching models in the domain of education and can be considered for a more general problem, educational resource matching. The educational resource matching finds educational resources to deliver customized information to users who use the education system [25]. Because our method tries to match a pair of questions and answers, it can be naturally extended to educational resource matching as well.

The rest of this paper is organized as follows. In Section 2, we describe related work. In Section 3, we describe the dataset used in this study. In Section 4, we present the proposed QA model. In Section 5, we present the performance evaluation. In Section 6, we conclude the paper and present the future work.

2 Related Work

2.1 Question-Answering Matching Models

Existing studies for the QA matching models can be largely classified into 1) QA pair matching [3,4], 2) machine reading-based models [5,6], and 3) KBQA [7–11]. First, for QA pair matching, Qiu et al. [4] proposed a convolutional neural tensor network (CNTN)-based model to extract the features of the sentence and found the most similar question based on the similarity of the extracted features. They used the English QA pair dataset provided by the Yahoo! search engine and the Chinese QA pair dataset provided by the Baidu Zhidao website and showed that the CNTN-based model outperformed the other models such as OKapi, TransLM, NBOW, and CNN. Cai et al. [3] presented a model combining bidirectional LSTM (BiLSTM) and CNN to detect whether two questions have the same intention. BiLSTM has a role to reflect the order of words in sentences into the model, and CNN has a role to extract the features of sentences. They presented that the proposed model combining BiLSTM and CNN outperformed a single CNN or BiLSTM model. Si et al. [26] matched similar questions using Word2vec and Glove to design a QA matching system in the financial field. They said that the performance of the matching model using Word2vec was the best as a result of the experiment. They said that the performance of the matching model using Word2vec was the best as a result of the experiment. Medved et al. [27] designed an automatic QA model using TF-IDF. Second, for machine reading-based QA models, Guven et al. [28] used natural language processing (NLP) models to improve the performance, especially when unrelated sentences are included in the dataset. They introduced three kinds of NLP models to select relevant sentences: 1) remove and compare (RC), 2) searching with named entity recognition (SNER), and 3) searching with part-of-speech (POS) tagging (SPOS). RC removes question terms and stopwords from each candidate sentence and answer, and then compares the two sentences to check if the selected sentence can be the answer. SNER finds question pronouns in question sentences and selects appropriate named entity recognition (NER) entity type, such as person, place, or time. Then, it determines if the selected sentence can be the answer based on the NER. SPOS removes the question terms from the selected sentence and parses it with the POS tag of the question terms. Then, it checks if the selected sentence is an answer. As a result, it improved the accuracy of QA matching by 6.6% to 8.76% compared to using BERT. Wadhwa et al. [6] conducted performance evaluation of 5 end-to-end machine learning models, i.e., 1) Bi-directional attention flow (BiDAF), 2) Gated self-matching networks (R-Net), 3) Document reader (DrQA), 4) Multi-paragraph reading comprehension (DocQA), and 5) Logistic regression (LR), to understand and compare their characteristics. They analyzed incorrect answers between models and suggested that an ensemble model could be effective to reduce incorrect answers. Stanford question answering dataset (SQUAD) [29] and Korean question answering dataset (KorQuad) [30] are datasets created to measure the performance of the machine reading-based QA model. Those datasets consist of the correct questions and answers, the documents with the answers, and the

locations of the answers in the documents. Third, for KBQA [7–11], some methods [7,8] constructed KB structures based on predefined templates for natural language questions and found answers based on the KB structures. These methods can be elaborated for predefined templates, however, it is difficult to respond to various types of questions except for the predefined ones. The other methods [9–11] constructed an embedded model for the KB structure through neural networks and matched the question with the answer based on the embedded model. Because it enables learning the relationship of questions through the embedded model, it can deal with various types of questions. Specifically, Jin et al. [10] presented the ComQA structure in which KBQA extracts answers based on semantic similarity to the knowledge base by decomposing complex questions into triple patterns. The overall steps are as follows: 1) identifying the entities, i.e., questions and their relationships, 2) transforming the syntactic patterns of the questions into a query graph to filter the topological mismatched patterns, and 3) consolidating knowledge-based information to rank the answers. Lai et al. [11] proposed a new lattice-based CNN model (LCNs) to improve the performance of QA models, especially when short texts with the same meaning were not properly interpreted due to word mismatch and diversity of expressions. The LCNs model, which receives a directed graph connected with neighboring words as an input, obtains feature vectors through several CNN kernels for capturing different context semantics. By performing CNNs over multiple n-gram contexts to exploit multi-granularity information, LCNs can effectively resolve the word mismatch problem compared to the existing KB word segmentation methods.

2.2 Text Classification Model

Recently, deep learning-based models have been proposed for text classification and improved classification accuracy compared to the traditional methods. Representative deep learning-based models include 1) LSTM, which considers the order of word appearances, and 2) CNN, which analyzes the features of words in a sentence. Zhang et al. [18] showed that the performance of individual LSTM and CNN models can be further improved by combining them, i.e., extracting the features of the sentence through CNN and understanding the order between the features through LSTM. They applied the combined model, LSTM-CNN, to the task of classifying the movie reviews according to subjectivity or objectivity. In order to produce a text binary classification model, Shin et al. [31] obtained a contrastive embedding model based on positive and negative text corpus, and showed that the proposed model performed better than the model using a single embedding model. BERT [17], a transformer-based bidirectional model, is a recent successful pre-trained model using the masked language model (MLM) that predicts empty words in a sentence and next sentence prediction (NSP) that learns whether two continuous sentences are correlated. Garrido Merchán et al. [32] showed that BERT outperformed the existing classification models such as voting classifier, logistic regression, and linear support vector classifier (SVC) in positive and negative classification on IMDB review dataset. The MLM-based dictionary training method has the disadvantage that it requires a lot of resources in the process of masking words in a sentence and predicting them. To overcome this limitation, Clark et al. [33] used replaced token detection (RTD), a pre-training method that uses a generator that fills in the masked words in a sentence and a discriminator that determines whether each word is the original or generated one. They showed that ELECTRA using RTD outperformed BERT. KoELECTRA, which was a pre-trained model using RTD for Korean, was also known to have better performance than other BERT-based models such as KoBERT, XLM-Roberta-Base, and HanBERT in the naver sentiment movie corpus (NSMC)[5] review classification model[6].

[5] https://github.com/e9t/nsmc.

[6] Park J., "Koelectra: Pretrained electra model for Korean," *GitHub repository.*, 2020.

2.3 Text Data Augmentation

The performance of the text classification model depends on the size and quality of the training dataset. In particular, when data in a specific class is not sufficient, classification performance is greatly reduced, which we call the data imbalance problem. In this case, data augmentation could be effective by increasing the datasets for the classes while maintaining the characteristics of the original datasets [20]. Various studies on text data augmentation have been conducted. Back-translation [21] uses two translators, where one translates the original sentence into another different language and the other retranslates the translated sentence back into the original language to produce another sentence with the same meaning. Contextual augmentation [34] removes the existing words in the original sentences and replaces them with the predicted words by language models such as bidirectional LSTM and CNN. EDA [20] consists of the following four augmentation strategies:1) Synonym replacement (SR): It randomly chooses n words from the sentence that are not stopwords. It replaces each of these words with one of its synonyms. 2) Random insertion (RI): It randomly chooses n words from the sentence that are not stopwords. It inserts the synonym for each of them into a random position in the sentence. 3) Random swap (RS): It randomly chooses two words in the sentence and swaps their positions. This is repeated n times. 4) Random deletion (RD): It randomly removes each word in the sentence with a probability p. In EDA, Wordnet [35] was used to define synonyms. n for SR, RI, and RS is obtained by a formula $n = \alpha l$, where l is the sentence length and α is a parameter that indicates the percent of the words that are changed in the sentence. For RD, $p = \alpha$ is used. Wei et al. [20] showed that the best performance of EDA was achieved when α was 0.1, and thus, we also followed the same value for α. They also showed that F1-Score becomes higher when EDA is applied to the classification using CNN and RNN models.

2.4 Sentence Embedding

The accurate measurement of the similarity between sentences, i.e., questions in QA models, significantly affects the performance of QA matching models. Methods for measuring the similarity between sentences are largely divided into 1) lexical matching and 2) semantic matching [36,37]. Lexical matching extracts the structural features of a sentence; Semantic matching interprets the semantics and relevance of sentences. Park et al. [37] applied a textual semantic matching between short texts (e.g., tweets) and long texts (e.g., news) based on various text similarity measures, 1) TF-IDF, 2) Word2Vec, 3) Doc2Vec, and 4) Rake, into the cyberattack prediction model. In this paper, we consider not only the sentence-based approach such as Doc2Vec [24] but also the word-based approach such as TF-IDF [38] and Word2Vec [23]. For the word-based approach, we transform the words or their features in the sentence into a vector space and obtain their averaged vector for embedding the sentence into the vector space. Tata et al. [39] proposed an approach for embedding the words in each sentence based on TF-IDF. Term frequency (TF) is the number of times a token appears in a sentence, indicating how important the token is in the sentence. Inverse document frequency (IDF) means the reciprocal of the number of sentences in which the token appears. TF-IDF can be obtained as the product of TF and IDF values. The TF-IDF value of each token is placed in the embedded space. They showed the effectiveness of TF-IDF-based embedding in choosing proper predicates by utilizing the cosine similarity of the embedded TF-IDF values of words in the sentence. Word2Vec, proposed by Mikolov et al. [23], learns the meaning of words through continuous bag of words (CBoW), which inserts words into a vector space and predicts target words based on surrounding words, or skip-gram, which predicts surrounding words from target words. Le et al. [24] proposed Doc2Vec to overcome ignoring the order between words and their meaning in the bag of words (BoW) model by introducing the concept of paragraph vectors. It learns to predict one word in a sentence using the paragraph vectors surrounding word vectors. In Doc2Vec, because it deals with a sentence itself in the vector space, we can directly utilize the vector of the sentence.

2.5 Summary

As described before, we focus on the following three key techniques in devising the proposed method: 1) text classification, 2) text data augmentation, and 3) sentence embedding. Table 1 shows the key techniques

that the existing studies for QA matching models utilize. As shown in Table 1, some existing studies utilized only sentence embedding, but none of them have not utilized the text classification and text data augmentation. In particular, we focus on adopting the text classification before QA matching and show its effectiveness. For the comparisons, we select two existing methods [26,27] that have not used the text classification while the used sentence embedding methods are the same as the proposed method.

Table 1: Summary for related work

Related work	QA pair matching				Machine reading		KBQA		Proposed method
	[3]	[4]	[26]	[27]	[28]	[6]	[9]	[10]	
Text classification [18,32]									O
Text data augmentation [20,21]									O
Sentence embedding [23,24,36,37]	O	O	O	O					O

3 Datasets

We used QA datasets collected from a mobile application for consulting entrance examinations and careers in South Korea[7]. It consists of 1) question datasets and 2) recommended answer datasets. The questions were labeled into seven major categories by question writers as follows: 1) study method, 2) mental management, 3) life management, 4) early entrance exam, 5) regular entrance exam, 6) middle schooler, and 7) career. One question can be mapped to at most one answer, i.e., some questions are not labeled; one answer can be mapped to multiple questions. Fig. 2 shows the distribution of the labeled question datasets. The rest of the categories except for regular and early entrance exams show a distinct separation. In the case of regular and early entrance exams, both are related to entrance exams, but we separate them because each of them has a distinct meaning in terms of the question writers.

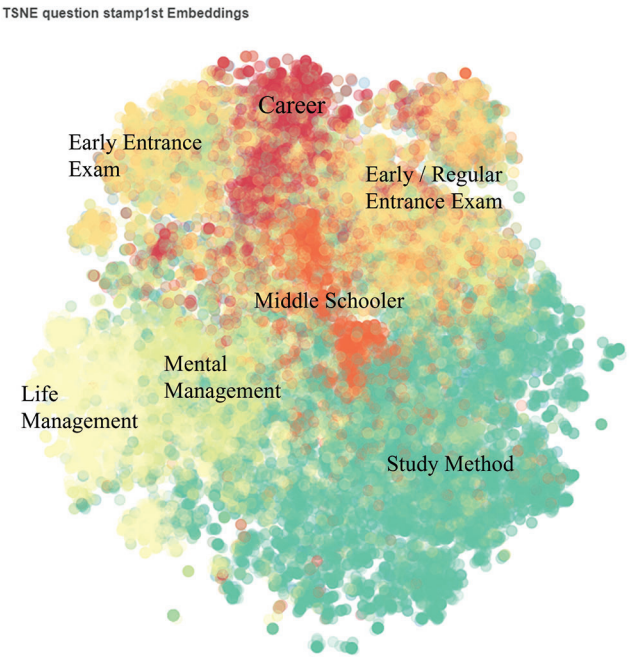

Figure 2: Distribution of question datasets

[7] https://flatgarden.kr/hakhak.

Table 2 describes the number of question datasets by category. The entire questions datasets of 28,634 are used to build the text classification model; among them, we used the QA pairs of 26,721 to validate the accuracy of the proposed model. The used datasets have the following distinguishing characteristics. First, the recommended answers for the questions are manually written. Second, because the questioner selects a category in advance before asking a question, the keywords used in the category could be missing in the questions themselves. To improve the quality of question datasets, we will introduce an augmentation technique, named meta data augmentation. We divide the entire datasets into the train and the test datasets by the ratio of 8:2. We observe a clear data imbalance: the data number of the largest class, "study method," is 9,989, and that of the smallest class, "regular entrance exam," is 825. Furthermore, we remove stopwords using the predefined Korean stopword list[8].

Table 2: Question datasets by each category

Class name	Train dataset	Test dataset	Total
Study method	7,977	2,012	9,989 (37%)
Early entrance exam	3,990	1,011	5,001 (19%)
Life management	2,792	670	3,462 (13%)
Mental management	2,690	649	3,339 (12%)
Career	1,642	437	2,079 (8%)
Middle schooler	1,643	383	2,026 (8%)
Regular entrance exam	653	172	825 (3%)
Total	21,387 (80%)	5,334 (20%)	26,721 (100%)

4 Proposed Model

4.1 Text Classification Model

Fig. 3 shows an architecture for the question classification model used in this paper. For this, we use five text classification models: LSTM-CNN, KoBERT, KoELECTRA, Multilingual BERT, and KR-BERT-MEDIUM. Recently, it has been known that pre-trained models perform well in natural language processing. Therefore, we tried to include most well-known pre-trained models that support the target language of the datasets, i.e., Korean. We also employed a fully trained model, LSTM-CNN, as the representative model to compare it with the pre-trained models. For LSTM-CNN, we fully train the model using the entire dataset. BERT and ELECTRA were usually known to show good performance as the pre-trained models. Therefore, we include the pre-trained models based on them while supporting Korean. For KoBERT, the Korean wiki dataset[9] is used for pre-training; for KoELECTRA, the news, Korean wiki, Namu wiki[10], and Everyone's corpus dataset[11] are used; for Multilingual BERT, Wikipedia data[12] obtained from the top 100 countries are used; for KR-BERT-MEDIUM, legal texts, Korean wikis, and news data crawled by the national law information center[13] are used. For BERT-based models, we conduct fine-tuning using our entire datasets based on pre-trained models. To improve the performance of the text classification model, we also adopt text data augmentation techniques, which will be described in Section 4.2.

[8] https://github.com/stopwords-iso/stopwords-ko.
[9] https://dumps.wikimedia.org/kowiki/.
[10] https://mu-star.net/wikidb.
[11] https://corpus.korean.go.kr/.
[12] https://meta.wikimedia.org/wiki/List_of_Wikipedias.
[13] https://www.law.go.kr/.

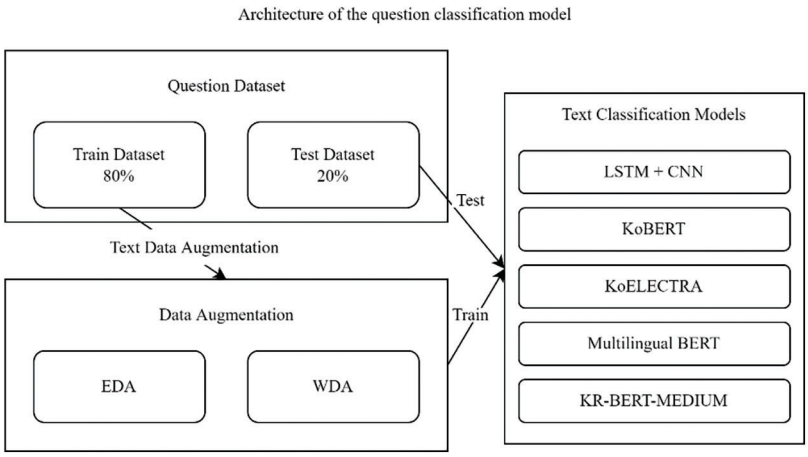

Figure 3: An architecture of the question classification model

4.2 Text Data Augmentation

For text data augmentation, although back-translation [21] showed better performance than EDA, the translator of back-translation misinterpreted major keywords related to the entrance exams and careers when we applied it into our dataset because the abbreviations for them are frequently used. Thus, we adopt SR in EDA, i.e., replacing the original words with synonyms, for text data augmentation. We use pre-defined Korean WordNet (KNN) [40] to define synonyms. Furthermore, we extend EDA based on Word2Vec, which we call WDA. To this end, specifically, we train Word2Vec using our datasets. Here, we use not only question datasets but also answer datasets to extend the trained model's coverage. To divide each sentence into words, we use a Korean morpheme analyzer, mecab-ko-dic[14]. Based on the trained Word2Vec, we replace each of the randomly chosen words in the sentence with the most similar word in Word2Vec.

4.3 Sentence Embedding Model

In this section, we devise an ensemble sentence embedding model by combining individual embedding models. Fig. 4 shows the concept of the presented ensemble sentence embedding model. Specifically, it considers various individual embedding models, considering sentence-based and word-based models. Then, it calculates the similarity scores based on the distance in the embedded vector space. By the performance evaluation for a given dataset, we choose appropriate individual models showing excellent performance. Then, we obtain a final similarity score by combining the similarity scores based on them. In order to detect bugs in the source code, Gharibi et al. [41] used a combined score for measuring bug localization including not only similarity measurement based on embedding models, i.e., TF-IDF and global vectors for word representation (GloVe), but also other problem-specific factors such as token matching, stack trace, and fixed bug report. In this paper, we extend the concept to combine multiple individual embedding models for QA matching problems, and consequently, this is the first effort for an ensemble embedding model for QA matching. We consider individual embedding models for sentences following the previous approaches in the QA matching problem [26,27]. The main difference between our model and previous models is that we apply the embedding models to each category classified by the text classification model, i.e., maintaining text classification models as many as the number of categories, while the previous models apply them to the entire dataset, i.e., constructing one text classification model. As the individual embedding models, we consider three kinds of models: 1) TF-IDF, 2) Word2Vec, and 3) Doc2Vec. That is, we consider not only the sentence-based model, i.e., Doc2Vec, but also the

[14] https://bitbucket.org/eunjeon/mecab-ko-dic/.

word-based models such as TF-IDF and Word2Vec. While we directly use the vector of the sentence in the sentence-based model, we obtain the averaged vector of words for each sentence based on each metric when we use the word-based models. For TF-IDF-based embedding, we train a TfidfVectorizer model. We fully train them using our entire question and answer datasets. To calculate the proximity in the embedded space, we use two kinds of metrics: 1) Euclidean distance and 2) cosine similarity. In Euclidean distance, the smaller the distance, the more similar questions are. In cosine similarity, the result is from 0 to 1, and the closer to 1, the more similar the two questions are.

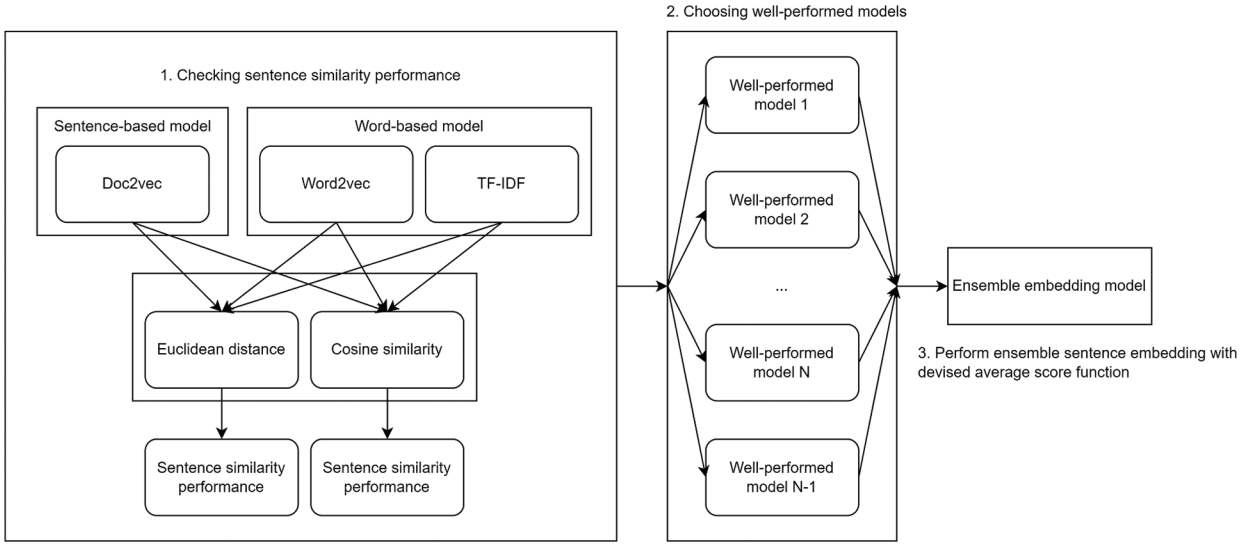

Figure 4: The concept of the ensemble sentence embedding model

4.4 Meta Data Augmentation

Considering the organization of QA datasets having categories, we present meta data augmentation to improve classification accuracy. That is, each question is assigned into multi-levels of categories that are specified by the question writers. For example, each question in our dataset consists of three-levels of categories. Each first-level category has 2∼6 second-level categories; again, each second-level category has 2∼6 third-level categories. For example, 'Study method' in the first-level category, has 'CSAT' and 'School grade' in the second-level category; 'CSAT' has 'Science', 'Korean', 'Society', 'Mathematics', 'English', 'Korean History', and 'Second Foreign Language' in the 3rd-level category. Considering the application's operations, the questions are written after selecting the first to third-level categories for them. To improve the performance of the text classification model, we utilize the second- and third-level category names as additional words for the corresponding question, which we call meta data augmentation. The insight for the meta data augmentation stems from the fact that, because each question is written after selecting the three-level of categories, the category name could be missing in the question itself. Because the order of words in the question does not affect calculating the similarity of questions, we maintain them as a list of words. The example of a question is as follows when the first, second, and third-level categories are "study methods", "CSAT", and "math", respectively: 1) (Original question): *I don't know what to do with the mock test.* 2) (word list for the original question): [*I, don't, know, what, to, do, with, the, mock, test*]. 3) (word list after meta data augmentation): [*CSAT, math, I, don't, know, what, to, do, with, the, mock, test*].

4.5 Overall Working Flow

Fig. 5 describes the working flow of our proposed QA pair matching model when a user question is given. The overall steps are as follows: 1) We determine a category of a given question through the text classification model. 2) We embed the question into the vector space for each category. 3) We compare the similarity in the embedding space between a given question and the question datasets in the category. 4) We extract N most similar questions to the given question. 5) We extract M answers from N questions in step 4 where each answer is connected to one question from the recommended answer datasets. Here, M can be different from N because an answer can be connected to multiple questions. Thus, N is determined so that M can meet the target number of answers (10 or 15 in the experiments).

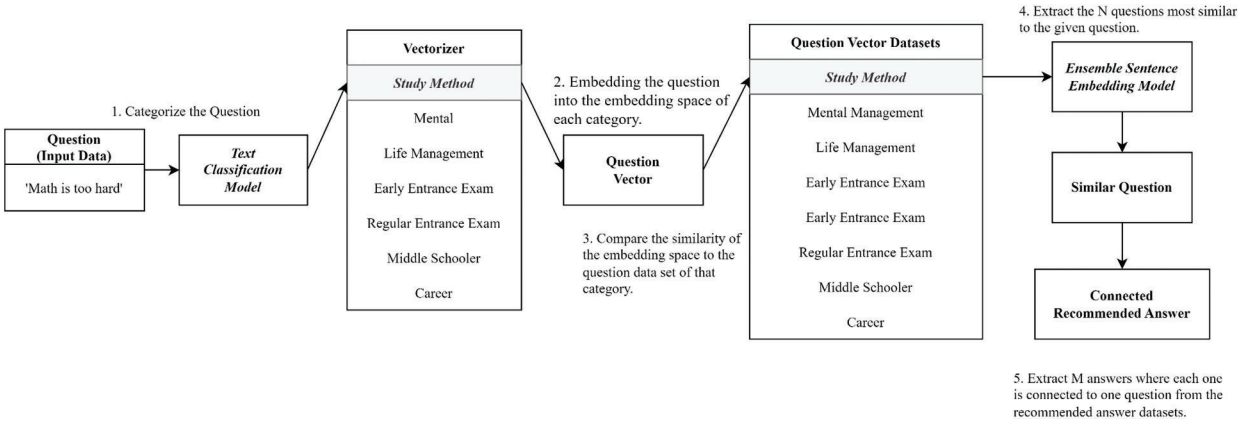

Figure 5: Overall working flow of the proposed QA pair matching model

5 Performance Evaluation

5.1 Experimental Methods and Environments

First, to measure the question classification performance, we use accuracy, precision, recall, and F1-score as the evaluation metrics. For this, we obtained four elements in the confusion matrix: True positive (TP), false positive (FP), false negative (FN), and true negative (TN). TP means that the model correctly predicted the class to which the question belongs; TN means that it correctly predicted the class to which the question does not belong; FP means that it incorrectly predicted the class to which the question belongs; FN means that it incorrectly predicted the class to which the question does not belong. Each metric is formalized as the following equations.

1) Accuracy: The ratio of the number of correct answers to the total number of predictions

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2) Precision: The ratio of correct prediction to the total number of true predictions

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

3) Recall: The ratio of positive predictions to the total number of correct predictions

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

4) F1-score: Harmonic average of precision and recall

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

Second, to evaluate the final accuracy of QA matching, we check if a ground truth answer for a given question is included in 10 or 15 answers that are predicted by the model. The experiments were conducted on a computer equipped with Intel Xeon CPU @ 2.30 GHz (Dual-Core), Nvidia Tesla K80, and Nvidia Tesla P100. Tensorflow version 2.8.2 and Keras 2.8.0 were used for LSTM-CNN; Pytorch version 1.12.0 along with cu113 was used for the other BERT-based models.

5.2 Hyper Parameter Setting

Table 3 shows the vocab sizes, parameter sizes, batch sizes, and epochs for the used text classification models. We determined the number of epochs for the question classification model right before overfitting occurs by checking the test accuracy. We determined the batch size of each classification model when it shows the best performance. The replacement probability p for words in EDA and WDA was set to 0.1, which was shown as the best performance in the previous study [20]. We used the cross entropy for the loss function, AdamW for the optimizer, and Cosine Lr for the scheduler. The ratio of training and testing datasets for the question classification model and sentence embedding model was 8:2. We used the Korean open-source morpheme analyzer, mecab-ko-dic, as the morpheme analyzer. The stopwords were removed in advance using the Korean stopword list. Doc2Vec was learned by the Distributed Memory version of Paragraph Vector (PV-DM) method; the vector size was set to 100, and the minimum frequency of words used for learning was set to 1. Word2Vec was learned by the CBoW method; the vector size was set to 100, the number of surrounding words used for CBoW was set to 5, and the minimum frequency of words used for learning was set to 5. For TF-IDF, the minimum frequency value of the document was set to 1, and the value was assigned to each word.

Table 3: Characteristics by text classification models

Models	LSTM-CNN	KoBERT	KoELECTRA-Base-v3	Multilingual BERT	KR-BERT-MEDIUM
Vocab	18,251	8,002	35,000	119,547	20,000
Parameter size	27,886,044	92,186,880	112,927,496	167,356,416	102,015,010
Batch size	100	64	2	2	2
Epoch	5	5	5	10	5

5.3 Experimental Results

5.3.1 Question Classification Model

Fig. 6 shows F1-score of question classification models. We confirm that all the models generally classify the questions with high accuracy. We indicate that the BERT-based pre-trained models generally outperform the fully trained model, LSTM-CNN. KoBERT showed the best performance, while LSTM-CNN was the worst.

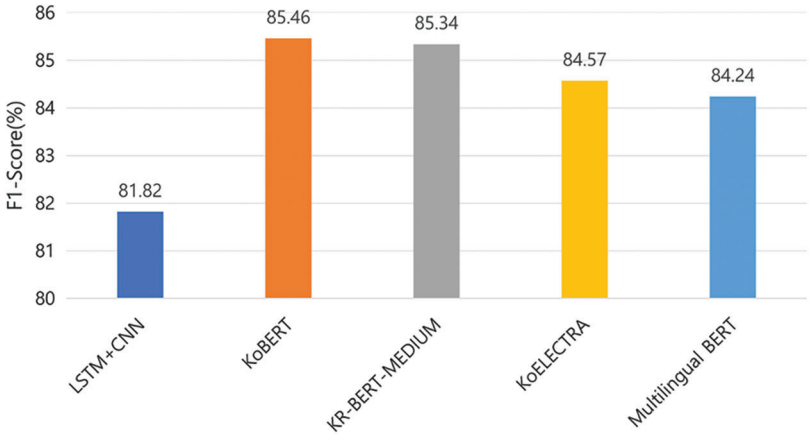

Figure 6: F1-score of question classification models

5.3.2 Data Augmentation

We augmented the number of questions for each category to 8,000 using EDA and WDA. Then, we compare the performance of using data augmentation techniques with that of using the original datasets. Table 4 shows the used datasets.

Table 4: Description of train dataset

Train dataset	Description
Default	Original dataset
EDA	Dataset in which the size of each class is augmented to 8000 through EDA
WDA	Dataset in which the size of each class is augmented to 8000 through WDA

Fig. 7 shows the effects of data augmentation by each model. We observe the clear performance improvement of data augmentation in LSTM-CNN, which is a fully trained model, in particular, when we use WDA, which is extended by this paper based on EDA and is fully trained using our datasets. However, the performance improvement of data augmentation is limited in other pre-trained models. It is natural because the data imbalance of the real dataset could be effectively resolved by the pre-trained models based on much larger datasets. Wei et al. [20] also reported that EDA was not effective in the pre-trained models. However, it is still worthwhile to confirm the effectiveness of data augmentation in the fully trained model when we can use only our datasets without pre-trained models.

5.3.3 Meta Data Augmentation

Fig. 8 shows the effects of the meta data augmentation as a sentence embedding model varies. Here, we retrieve 15 recommended answers and compare two cases where the one applies the meta data augmentation to for each model and the other does not. The result shows the clear performance improvement by the meta data augmentation. When Doc2Vec is used, the overall performance degrades compared to when Word2Vec and TF-IDF are used, and in this case, the meta data augmentation is not also effective. However, in both Word2Vec and TF-IDF, the meta data augmentation significantly improves the model performance, i.e., by 4.65%~5.42%.

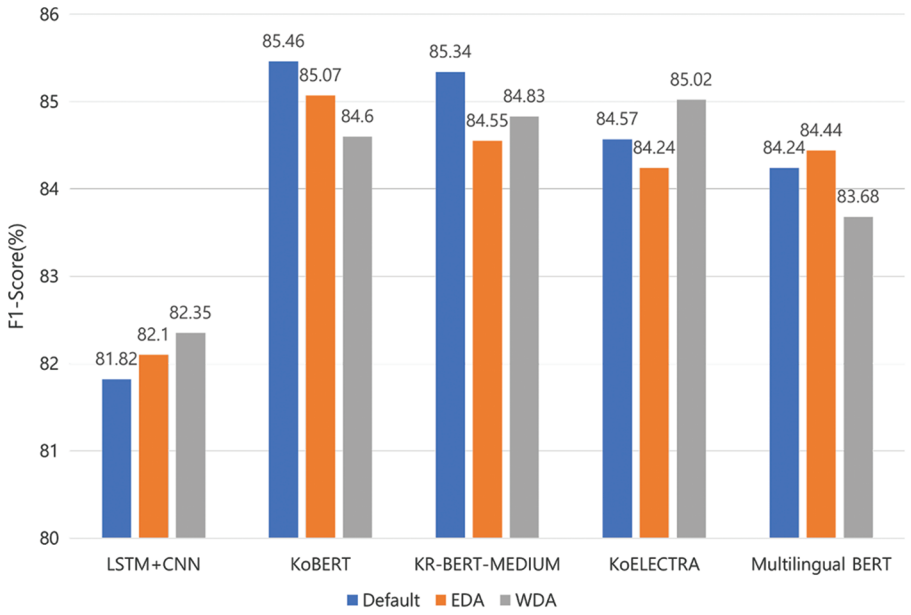

Figure 7: F1-Score according to data augmentations

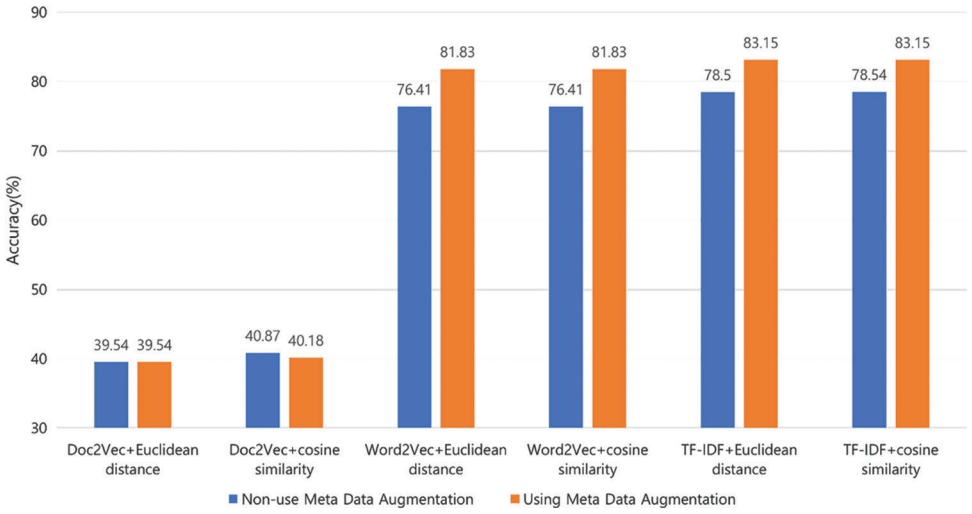

Figure 8: Performance comparison by meta data augmentation

5.3.4 Sentence Embedding Model

Fig. 9 shows the performance comparison of our proposed model as we vary the sentence embedding models, especially to check the effectiveness of our ensemble sentence embedding model. Here, we retrieved 10 or 15 answers recommended by each model for a given question. First, in individual embedding models, TF-IDF and Word2Vec outperform Doc2Vec, and they are comparable to each other. For the similarity measurement, cosine similarity outperforms Euclidean distance in all the embedding models.

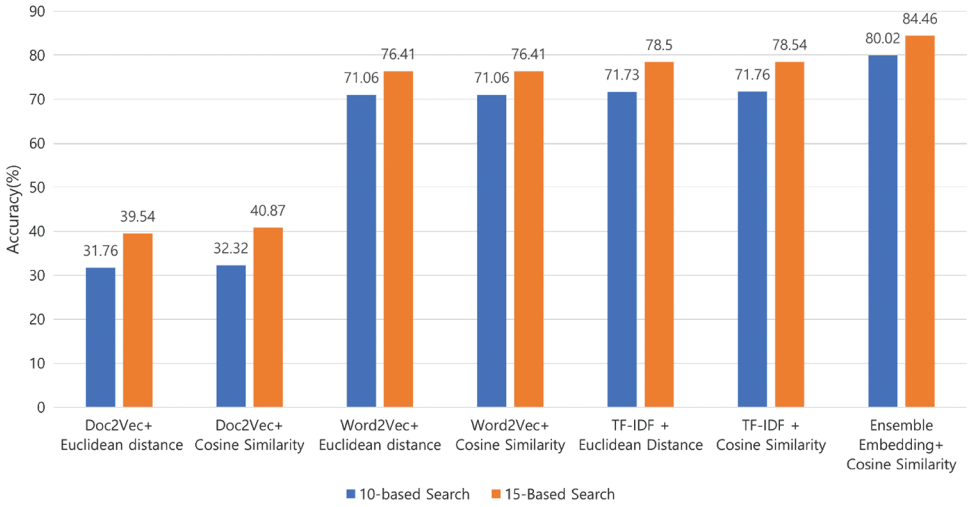

Figure 9: Performance comparison by the sentence embedding models

These results provide the basis to choose the models and similarity measurement to design the ensemble sentence embedding model so as to further improve each single model. Based on the experimental results for individual models and similarity measurements, we used the two sentence embedding models, TF-IDF and Word2Vec, for the ensemble sentence embedding model. In addition, we use cosine similarity as the similarity measurement. We note that the result shows that the ensemble sentence embedding model improves the accuracy of Word2Vec and TF-IDF-based embedding by 8.05%~8.96% and 5.92%~8.26%, respectively.

5.3.5 Comparison with the Existing Methods

In comparison with the existing methods, we focused on the effects of the classification model-based QA matching models because our approach can be applied into any other advanced embedding models. As a result, we selected two comparisons [26,27] that did not use the classification models while using the same embedding models as used in this paper. The existing QA matching models [26,27] extract the most similar questions from the entire datasets, while we adopt the text classification model to select the most relevant category first. Fig. 10 shows the comparison results of the proposed model with the existing models using various embedding models: 1) Word2Vec [26] and 2) TF-IDF [27]. Finally, our ensemble model significantly outperforms the existing Word2Vec-based model [26] and TF-IDF-based model [27] by 13.08%~14.16% and by 9.38%~11.46%, respectively.

Fig. 11 compares the inference speed of the proposed model with those of the existing models for answering the entire 21,629 questions. We also confirm that the time to extract answers using our model is faster than using the existing models, about 2.61 to 5.07 times. This stems from the fact that our model effectively reduces the search space for a given question due to the text classification model.

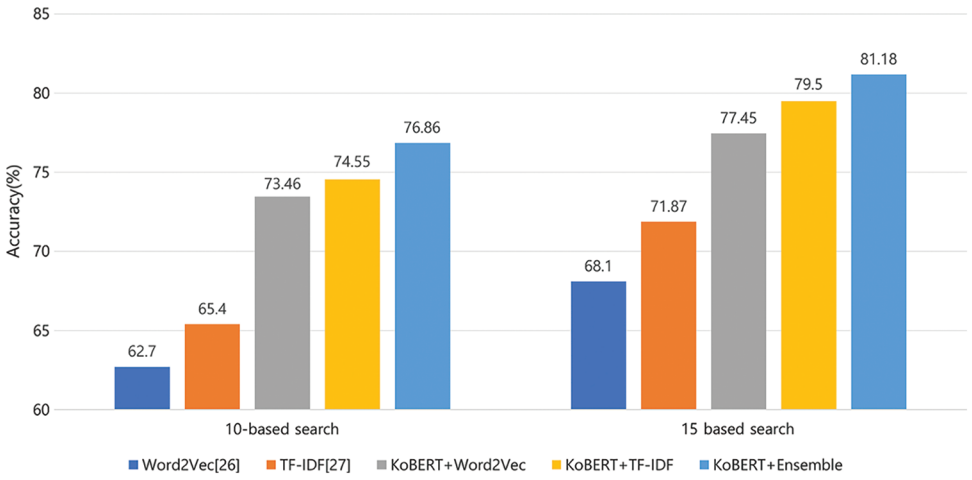

Figure 10: Performance comparison with the existing models

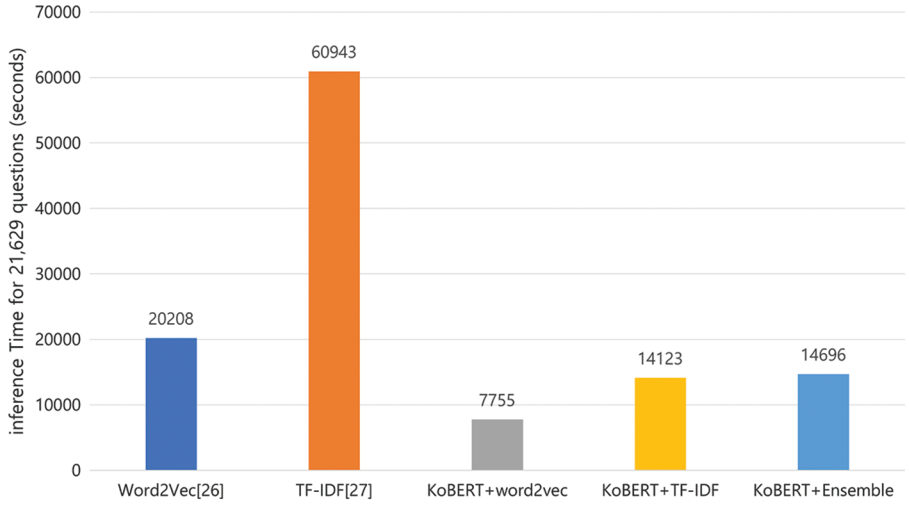

Figure 11: Comparison of the model inference time with the existing models

6 Conclusions

In this paper, we proposed a QA pair matching model based on text classification to overcome the limitations that the existing QA pair matching systems are time consuming and less accurate because they tried to find the recommended answers from the entire datasets without the classification of target datasets. By incorporating the text classification model, we can effectively reduce the search space for finding the most relevant questions to a given question. As a result, the proposed model significantly improved the accuracy of QA matching and dramatically reduced the model inference time. Furthermore, to improve the performance of finding similar questions in each category, we presented an ensemble sentence embedding model by combining well-performed individual models, indicating the performance improvement compared to all the individual embedding models. We evaluated the performance of the proposed QA pair matching model. As a result, we showed the effectiveness of each component in the proposed model and confirmed that the proposed model outperformed the existing QA matching models. In this paper, using the real-world datasets collected from the mobile application in service, we confirmed the actual effectiveness of each component and our final QA matching model compared to the existing

models in terms of both accuracy and inference speed. However, the proposed model has the inherent constraints that require the predefined categories on question and answering datasets, accelerating the effects of the classification model in the process. Therefore, to effectively handle the limitations in terms of the practical viewpoints, we plan to devise unsupervised learning-based techniques through clustering or semi-supervised learning models. Furthermore, we plan to propose the optimal method for each component. That is, for data augmentation, we can consider a semi-supervised technique that learns unlabeled data. To extend the embedding models, we can consider GloVe, Sentence-BERT, and Sentence-Transformers.

Funding Statement: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1F1A1067008), and by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2019R1A6A1A03032119).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] E. Sneiders, "Automated question answering using question templates that cover the conceptual model of the database," in *Int. Conf. on Application of Natural Language to Information Systems*, Berlin, Heidelberg, Springer, pp. 235–239, 2002.

[2] W. T. Yih and H. Ma, "Question answering with knowledge base, Web and beyond," in *Proc. of the 39th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, New York, NY, USA, pp. 1219–1221, 2016.

[3] L. Q. Cai, M. Wei, S. T. Zhou and X. Yan, "Intelligent question answering in restricted domains using deep learning and question pair matching," *IEEE Access*, vol. 8, pp. 32922–32934, 2020.

[4] X. Qiu and X. Huang, "Convolutional neural tensor network architecture for community-based question answering," in *Twenty-Fourth Int. Joint Conf. on Artificial Intelligence*, Buenos Aires, Argentina, pp. 1305–1311, 2015.

[5] B. Bi, C. Wu, M. Yan, W. Wang and J. Xia *et al.,* "Incorporating external knowledge into machine reading for generative question answering," arXiv preprint arXiv:1909.02745, 2019.

[6] S. Wadhwa, K. R. Chandu and E. Nyberg, "Comparative analysis of neural QA models on SQuAD," arXiv preprint arXiv:1806.06972, 2018.

[7] L. S. Zettlemoyer and M. Collins, "Learning context-dependent mappings from sentences to logical form," in *Proc. of the Joint Conf. of the 47th Annual Meeting of the ACL and the 4th Int. Joint Conf. on Natural Language Processing of the AFNLP*, Suntec, Singapore, vol. 2, pp. 976–984, 2009.

[8] C. Unger, L. Bühmann, J. Lehmann, A. C. Ngonga Ngomo and D. Gerber *et al.,* "Template-based question answering over RDF data," in *Proc. of the 21st Int. Conf. on World Wide Web*, New York, NY, USA, pp. 639–648, 2012.

[9] Y. Lan, S. Wang and J. Jiang, "Knowledge base question answering with a matching-aggregation model and question-specific contextual relations," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 10, pp. 1629–1638, 2019.

[10] H. Jin, Y. Luo, C. Gao, X. Tang and P. Yuan, "ComQA: Question answering over knowledge base via semantic matching," *IEEE Access*, vol. 7, no. 18749143, pp. 75235–75246, 2019.

[11] Y. Lai, Y. Feng, X. Yu, Z. Wang, K. Xu *et al.,* "Lattice cnns for matching based Chinese question answering," in *Proc. of the AAAI Conf. on Artificial Intelligence*, California, USA, vol. 33, no. 1, pp. 6634–6641, 2019.

[12] D. Zhang and W. S. Lee, "Question classification using support vector machines," in *Proc. of the 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, New York, NY, USA, pp. 26–32, 2003.

[13] M. Ikonomakis, S. Kotsiantis and V. Tampakas, "Text classification using machine learning techniques," *WSEAS Transactions on Computers*, vol. 4, no. 8, pp. 966–974, 2005.

[14] I. Rish, "An empirical study of the naive Bayes classifier," in *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, Washington, USA, vol. 3, no. 22, pp. 41–46, 2001.

[15] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 Int. Conf. on Engineering and Technology (ICET)*, Antalya, Turkey, pp. 1–6, 2017.

[16] P. Liu, X. Qiu and X. Huang, "Recurrent neural network for text classification with multi-task learning," arXiv preprint arXiv:1605.05101, 2016.

[17] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota, 2019.

[18] J. Zhang, Y. Li, J. Tian and T. Li, "LSTM-CNN hybrid model for text classification," in *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conf. (IAEAC)*, Antalya, Turkey, pp. 1675–1680, 2018.

[19] C. Shorten, T. M. Khoshgoftaar and B. Furht, "Text data augmentation for deep learning," *Journal of big Data*, vol. 8, no. 1, pp. 1–34, 2021.

[20] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks," arXiv preprint arXiv:1901.11196, 2019.

[21] S. Edunov, M. Ott, M. Auli and D. Grangier, "Understanding back-translation at scale," arXiv preprint arXiv:1808.09381, 2018.

[22] J. Ramos, "Using tf-idf to determine word relevance in document queries," in *Proc. of the First Instructional Conf. on Machine Learning*, California, USA, vol. 242, no. 1, pp. 29–48, 2003.

[23] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[24] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. of the 31st Int. Conf. on Machine Learning*, Bejing, China, vol. 32, no. 2, pp. 1188–1196, 2014.

[25] C. Z. Xiang, N. X. Fu and T. R. Gadekallu, "Design of resource matching model of intelligent education system based on machine learning," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 9, no. 6, pp. e43–e43, 2022.

[26] S. Si, W. Zheng, L. Zhou and M. Zhang, "Sentence similarity computation in question answering robot," *Journal of Physics: Conference Series*, vol. 1237, no. 2, pp. 022093, 2019.

[27] M. Medved and A. Horák, "Sentence and word embedding employed in open question-answering," *ICAART*, vol. 2, pp. 486–492, 2018.

[28] Z. A. Guven and M. O. Unalir, "Natural language based analysis of SQuAD: An analytical approach for BERT," *Expert Systems with Applications*, vol. 192, pp. 116592, 2022.

[29] P. Rajpurkar, J. Zhang, K. Lopyrev and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," arXiv preprint arXiv:1606.05250, 2016.

[30] S. Lim, M. Kim and J. Lee, "Korquad1. 0: Korean qa dataset for machine reading comprehension," arXiv preprint arXiv:1909.07005, 2019.

[31] H. S. Shin, H. Y. Kwon and S. J. Ryu, "A new text classification model based on contrastive word embedding for detecting cybersecurity intelligence in twitter," *Electronics*, vol. 9, no. 9, pp. 1527, 2020.

[32] E. C. Garrido-Merchán and S. Gonzalez-Carvajal, "Comparing BERT against traditional machine learning text classification," arXiv preprint arXiv:2005.13012, 2020.

[33] K. Clark, M. T. Luong, Q. V. Le and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," arXiv preprint arXiv:2003.10555, 2020.

[34] S. Kobayashi, "Contextual augmentation: Data augmentation by words with paradigmatic relations," arXiv preprint arXiv:1805.06201, 2018.

[35] G. A. Miller, "WordNet: A lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[36] P. Achananuparp, X. Hu and X. Shen, "The evaluation of sentence similarity measures," in *Int. Conf. on Data Warehousing and Knowledge Discovery*, Berlin, Heidelberg, Springer, pp. 305–316, 2008.

[37] J. H. Park and H. Y. Kwon, "Cyberattack detection model using community detection and text analysis on social media," *ICT Express*, 2021. https://doi.org/10.1016/j.icte.2021.12.003

[38] A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, 2003.

[39] S. Tata and J. M. Patel, "Estimating the selectivity of tf-idf based cosine similarity predicates," *ACM Sigmod Record*, vol. 36, no. 2, pp. 7–12, 2007.

[40] A. S. Yoon, S. H. Hwang, E. R. Lee and H. C. Kwon, "Construction of Korean WordNet," *Journal of KIISE: Software and Applications*, vol. 36, no. 1, pp. 92–108, 2009.

[41] R. Gharibi, A. H. Rasekh, M. H. Sadreddini and S. M. Fakhrahmad, "Leveraging textual properties of bug reports to localize relevant source files," *Information Processing & Management*, vol. 54, no. 6, pp. 1058–1076, 2018.